Scheduling coupled tasks with time windows: a parameterized complexity analysis

Maher Mallem, Claire C. Hanen, Alix Munier-Kordon

Scheduling coupled tasks with time windows: a parameterized complexity analysis

Maher Mallem[1][0000−0001−5654−1090], Claire Hanen[1,2][0000−0003−2482−5042], and Alix Munier Kordon[1][0000−0002−2170−6366]

[1] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
{Maher.Mallem,Claire.Hanen,Alix.Munier}@lip6.fr
http://www.lip6.fr
[2] UPL, Université Paris Nanterre, F-92000 Nanterre,France

Abstract. This paper studies the parameterized complexity of coupled tasks scheduling problems with time windows considering two parameters: the pathwidth and the slack. Three types of precedence delays are studied: minimum, maximum and exact. For all cases we prove that these problems are para-NP-hard parameterized by the slack even for unit processing time jobs. We exhibit a relation between the slack and the pathwidth that allows us to extend these results for the pathwidth parameter. Lastly a polynomial procedure is provided in the special case where the slack is equal to one.

Keywords: parameterized complexity · scheduling · coupled tasks

1 Introduction

Since the seventies many studies have been devoted to scheduling problems on a single processor with different settings and optimization criteria. If we just consider decision problems only few problems can be solved in polynomial time - like scheduling unit execution time jobs with time windows [17] or scheduling jobs with precedence constraints and deadlines [14]. On the other hand many sequencing problems with simple settings are NP-hard in the strong sense, especially when time windows are considered [15].

This paper addresses a single machine problem with time windows and precedence constraints with delays. We are given a set of jobs $\mathcal{T}$. Each job $J \in \mathcal{T}$ is characterized by a processing time $p(J) = 1$ and a time window $[r(J), d(J))$ in which it should be scheduled. A non negative delay ℓ_a is attached to each precedence constraint a between two jobs J and J'. Three types of delays are considered. In the case of minimum delays, the difference between the starting time of J' and the completion time of J must not be less than ℓ_a; in the case of maximum delays, this difference must be not greater than ℓ_a; for exact delays this difference must be exactly equal to ℓ_a.

We study an interesting special case called coupled tasks where the precedence graph is just a set of disjoint arcs. They have been studied for many years due to their many applications. Scheduling jobs on single machine assuming unit

processing times and coupled tasks has been proven NP-hard by Yu [20] for exact and minimum delays. See [5, 13] for a survey of complexity results of coupled tasks with exact delays.

In order to go beyond these results parameterized complexity gives us tools to refine the analysis and understand deeper on which parameter of the instances the problem complexity relies. Given a parameter k and denoting n the input size, a problem is called fixed-parameter tractable (FPT) with respect to parameter k if it can be solved in time $poly(n) \times f(k)$ with f an arbitrary function [6].

When the studied problem is believed to not be FPT, the para-NP class is used as a parameterized version of NP: a problem is in para-NP with respect to parameter k if there is a non-deterministic algorithm that solves it in time $poly(n) \times f(k)$ with f an arbitrary function. In order to prove that a problem is para-NP-hard with respect to k, it is enough to prove that the un-parameterized problem is NP-hard for some fixed value of k [8]. The W-hierarchy defined in [7] is an additional widely used tool in parameterized complexity. It is a sequence of intermediate complexity classes between FPT and para-NP, which allows us to further investigate the time complexity of a parameterized problem.

Several parameters have been considered for the analysis of the parameterized complexity of scheduling problems, like the maximum processing time $p_{\max}$ [18] or the width of the precedence graph [3]. Considering precedence delays we can first mention the work of Bessy et al. [2] about coupled tasks with due dates, a compatibility graph and a common deadline on a single machine. Their parameter was the number of jobs ending before their due date. They established W[1]-hardness for this problem and developed a FPT algorithm when the maximum duration of a job (i.e. the processing time of the two tasks plus their delay) is bounded. Van Bevern et al. [3] proved that the problem of scheduling a precedence graph on two machines with processing times in $\{1, 2\}$ is W[2]-hard with respect to the width. In the same paper they introduced an additional parameter λ that measures the maximum allowed lag of a job with respect to its earliest start time (ignoring resource constraints). They proved that the Resource-Constrained Project Scheduling Problem (RCPSP) is FPT parameterized by both the allowed lag and the width.

In the presence of time windows, several authors considered parameters taking into account their maximum overlapping number. Bodlaender and van der Wegen [4] addressed the single-machine scheduling of a set of chain like precedence constraints with delays where the time windows are expressed on chains and not on individual jobs. Their parameter - called the thickness - corresponds to the maximum number of overlapping chains. They proved that the problem with minimum delays given in unary is W[2]-hard with respect to the thickness.

Similarly the pathwidth μ is the maximum number of overlapping time windows minus one. Munier-Kordon [19] developed a FPT algorithm for the decision problem of scheduling unit execution time jobs with precedence constraints and time windows of jobs parameterized by μ. In [16] scheduling chains of unit execution time jobs with minimum or exact delays on a single processor was proved to be para-NP-hard with parameter μ, whereas it was proved to be FPT if no

precedence delays are assumed in [10]. In addition to pathwidth μ the slack σ was considered by several authors as an extra property around time windows. The slack corresponds to the maximum number of starting times for a job minus one. A FPT with the tuple of parameters "slack and pathwidth" (or "maximum processing time and pathwidth") was proposed in [11] for a parallel machine scheduling problem with precedence and time windows. For a single machine problem with job rejection criteria a FPT with parameters slack and pathwidth has also been proposed [1].

The aim of this paper is to analyze the parameterized complexity of a single machine scheduling unit execution time jobs with precedence delays and time windows parameterized by the slack alone. To the best of our knowledge this parameter has not been investigated for this problem yet. In fact our goal is to establish as accurately as possible the boundary between the FPT problems and the others. Moreover we include the case of maximum delays in our study, which is usually less studied than minimum and exact delays.

Our contribution first establishes a relation between the slack and the pathwidth, leading to the following property: if a problem is FPT parameterized by the pathwidth then it is also FPT for the slack. Conversely if it appears to be NP-hard for a bounded slack, then it is for a bounded pathwidth. This implies that results already known for the pathwidth can be transferred to the slack. In Section 2 we present our notations, the parameters and the relation between slack and pathwidth.

Then we show in Section 3 that scheduling coupled tasks with unit execution time on a single machine with minimum, maximum or exact delays is para-NP-hard with respect to the slack by establishing the NP-hardness of these problems with slack value $\sigma = 2$. In Section 4 we provide a polynomial algorithm for the problem with general precedence graph and slack value bounded by 1 by showing an equivalence to the 2-SAT problem. Finally in Section 5 we draw some perspectives.

2 Definitions and parameters

2.1 Preliminary definitions

In this paper we study the problem denoted by $1|prec(\ell_{i,j}), p_j = 1, r_j, d_j|\star$ in the standard notation of Graham [9]. An instance of this problem is characterized by a set $\mathcal{T}$ of n unit execution time jobs. Each job J in $\mathcal{T}$ has a processing time $p(J) = 1$, a release date $r(J)$ and a deadline $d(J)$. Moreover we consider an acyclic precedence graph *prec* representing precedence relations between the jobs in $\mathcal{T}$. Each arc $a = (J, J')$ in *prec* is valued by a nonnegative integer ℓ_a. A schedule s defines for each job J a starting time $s(J)$ so that $r(J) \leq s(J) < d(J)$ and for any arc $a = (J, J')$ in *prec*:

minimum delays: $s(J') \geq s(J) + 1 + \ell_a$
maximum delays: $s(J') \leq s(J) + 1 + \ell_a$
exact delays: $s(J') = s(J) + 1 + \ell_a$

Definition 1 (Proper delay). *Given an arc $a = (J, J')$ the delay ℓ_a is called proper when $\ell = r(J') - r(J) - 1$.*

Lemma 1. *In any feasible schedule if a job J is scheduled at time $r(J)+k, k \geq 0$ and there is a proper delay ℓ between J and another job J', then J' must be scheduled at time:*

1. *$r(J') + k$ or later if ℓ is a minimum delay,*
2. *$r(J') + k$ or earlier if ℓ is a maximum delay,*
3. *exactly $r(J') + k$ if ℓ is an exact delay.*

Proof. Let s be a feasible schedule.

1. With minimum delays we have: $s(J') \geq s(J) + 1 + \ell$.
   With $s(J) = r(J) + k$ and $\ell = r(J') - r(J) - 1$ we have:
   $s(J') \geq [r(J) + k] + 1 + [r(J') - r(J) - 1] = r(J') + k$.
2. and 3. are proved the same way. □

A large part of this paper is devoted to the special case of coupled tasks. A coupled task is a set of two unit execution time jobs linked by an arc of the precedence graph. So a coupled task can be described by a triplet (J_1, ℓ, J_2). In scheduling problems with coupled tasks we assume that each job has its time window and that no precedence constraint exists between different coupled tasks.

Definition 2 (Properly coupled task). *A coupled task (J_1, ℓ, J_2) with the same time window length $\alpha = d(J_1) - r(J_1) = d(J_2) - r(J_2)$ is called properly coupled if the delay ℓ between both jobs is proper. Then this coupled task can be represented by the triple $\langle r(J_1), r(J_2), \alpha \rangle$.*

We now introduce a notion that will be used in the complexity reductions.

Definition 3 (Loop of properly coupled tasks). *A loop of properly coupled tasks for minimum delays (resp. maximum delays) is composed by two set of properly coupled tasks with time windows of length 2; set $\mathcal{F}$ with f coupled task is called the forward phase of the loop, and set $\mathcal{B}$ with b coupled tasks its backward phase. $\mathcal{F} = \{\langle r(F(i)), r(F'(i)), 2\rangle, i \in \{1, \ldots, f\}\}$ $\mathcal{B} = \{\langle r(B(i)), r(B'(i)), 2\rangle, i \in \{1, \ldots, b\}\}$. The sets have the following properties:*

- *Time windows of two consecutive forward tasks intersect: $\forall i \in \{1, \ldots, f-1\}$, $r(F(i+1)) = r(F'(i)) + 1$ (resp. $r(F(i+1)) = r(F'(i)) - 1$);*
- *Similarly, $\forall i \in \{1, \ldots, b-1\}$, $r(B(i+1)) = r(B'(i)) + 1$ (resp. $r(B(i+1)) = r(B'(i)) - 1$);*
- *The first and the last time windows of forward and backward phase coincide: $r(F(1)) = r(B(1))$ and $r(F'(f)) = r(B'(b))$.*

Now looking at Figure 1 which depicts a loop with two forward tasks and one backward task, the scheduled time of $F(1)$ is accurately propagated throughout the whole loop with minimum delays no matter if it is scheduled left or right. Thus a loop can be seen as an *on/off switch* which can be set by any job in the loop. This property is extended to any loop in Lemma 2.

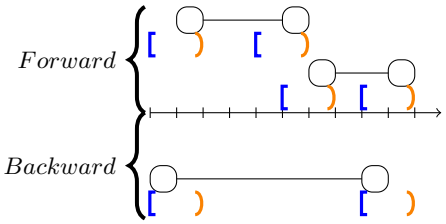

Fig. 1. A loop of properly coupled tasks for minimum delays.

Lemma 2. *A loop of coupled tasks with minimum (resp. maximum) delays has exactly two feasible schedules. The first one is the schedule where every job of $\mathcal{F}$ is scheduled to the right (resp. the left) of its time window while every job of coupled tasks of $\mathcal{B}$ is scheduled to the left (resp. the right) of its time window. Then we say that the loop is ON. The second one is where all jobs in and $\mathcal{F}$ and $\mathcal{B}$ are scheduled the other way around. Then we say that the loop is OFF.*

Proof. We prove the lemma in the minimum delays case (similar arguments hold for the maximum/exact delays case). Notice that for any coupled task of $\mathcal{F}$ if the second job of the coupled task is performed to the right of its time window at $r(F'(i)) + 1$ then the first job of the next coupled task can only be performed at $r(F(i+1)) + 1$. So this "right-shift" information is propagated along $\mathcal{F}$ until the last forward job, which is also executed at the right of its time window. Similarly if the second job is performed to the left of its time window at $r(F'(i))$ then the first job of the same coupled tasks is performed to the left of its interval at $r(F(i))$. Now this implies that the previous coupled task is also performed to the left, at $r(F'(i-1))$, and this can be propagated to the left. The same argument holds for $\mathcal{B}$.

Now since the first time window of length 2 is common to the first job of both $\mathcal{F}$ and $\mathcal{B}$, there can be only two possibilities: either the first job of $\mathcal{B}$ starts first and then the first job of $\mathcal{F}$ is scheduled, or the reverse case. In the first case, since we have minimum delays, this implies that all jobs of $\mathcal{F}$ are scheduled to the right of their time window. Now in the last time window of both chains, if the job $F'(f)$ of $\mathcal{F}$ is scheduled to the right, the job $B'(b)$ of $\mathcal{B}$ that shares the same interval is scheduled to the left of its time window, and so are all the coupled tasks of $\mathcal{B}$. Hence we are in the ON state of the loop. A symmetric argument holds for the OFF state. □

2.2 Slack vs pathwidth

Let us now recall the definition of the two parameters we use in this paper for our parameterized complexity analysis:

- pathwidth: at most $\mu + 1$ job time windows intersect at each time t.

– slack: for all jobs $J \in \mathcal{T}$ the number of starting times minus one is given by $d(J) - p(J) - r(J)$, so we have: $d(J) - p(J) - r(J) \leq \sigma$.

By using these definitions a relation between both parameters in feasible schedules can be established.

Theorem 1. *If a single machine scheduling instance with time windows is feasible, then* $\mu \leq 2\sigma$.

Proof. By the definition of slack σ:

$$\begin{cases} r(J) \geq d(J) - p(J) - \sigma \\ d(J) \leq r(J) + p(J) + \sigma \end{cases} \tag{1}$$

Let J be a job and t be a time slot. If t is part of interval $[r(J), d(J))$ then:

$$\begin{cases} t < d(J) \\ t \geq r(J) \end{cases} \tag{2}$$

Then by using both inequalities on the definition of slack σ we get:

$$\begin{cases} r(J) > t - p(J) - \sigma \\ d(J) \leq t + p(J) + \sigma \end{cases} \tag{3}$$

This means that whenever job J is scheduled, $p(J)$ consecutive time slots will be taken by J in time interval $[t - \sigma - p(J) + 1, t + \sigma + p(J))$. Thus at least one time unit will be taken by J in time interval $[t - \sigma, t + \sigma + 1)$. Since we only have one machine, this means that at most $2\sigma + 1$ jobs can have t be a part of their time window $[r(J), d(J))$ in any feasible schedule of this instance. This yields the wanted inequality: $\mu \leq 2\sigma$. □

3 Hardness reductions with with $\sigma = 2$

In this section we prove para-NP-completeness of our three single machine scheduling problems parameterized by slack σ. Then we infer para-NP-completeness of the variants with pathwidth μ as the parameter.

We use the following result from Flum and Grohe's book [8]: *if a (nontrivial) problem* $\mathcal{P}$ *is* NP*-complete with a fixed value of some parameter* k*, then the parameterized problem* $(\mathcal{P}, k)$ *is para-*NP*-complete.* For all three delay types $X \in \{min, max, ex\}$ we prove the scheduling problem $1|(1, \ell^X, 1), r_j, d_j|\star$ to be NP-complete with $\sigma = 2$.

The three problems are all trivially in NP by guessing the starting time of each job then checking if this leads to a feasible schedule. For the hardness proofs all reductions will start from the (strongly) NP-hard 3-Coloring graph problem [12]. Let $G = (V, E)$ be the input graph (with no self-loops). Let $v_0, \ldots, v_{n-1}$ be the vertices in V and $e_0, \ldots, e_{m-1}$ be the edges in E. Let $n = |V|$ and $m = |E|$. The colors will be named 0, 1 and 2.

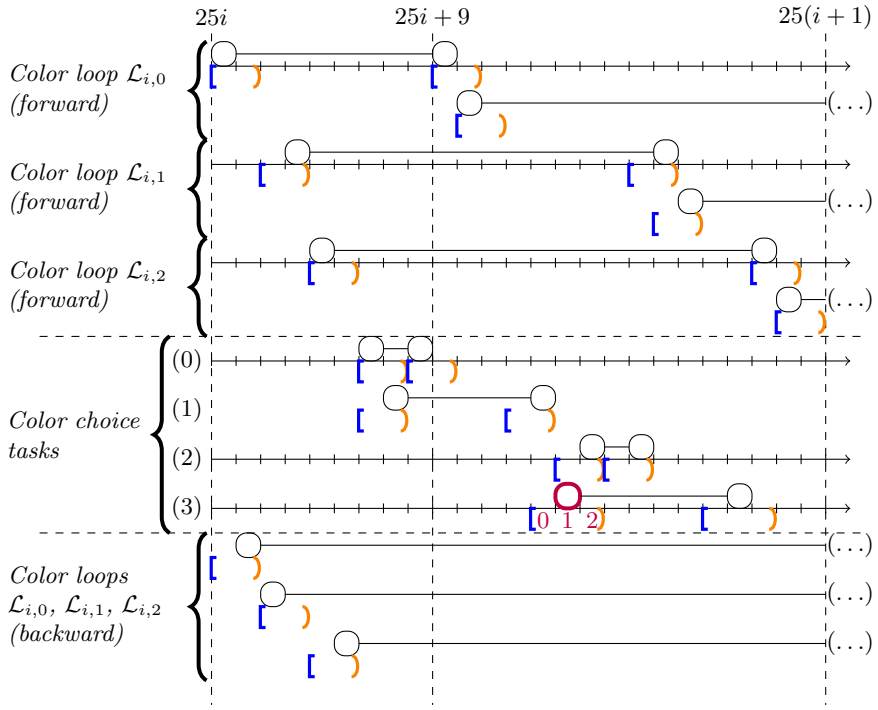

Fig. 2. The color choice segment corresponding to node v_i in the reduction with minimum delays. Color loop $\mathcal{L}_{i,1}$ is ON and the other two are OFF.

3.1 Outline of the reductions

For every delay type $X \in \{min, max, ex\}$ an instance of $1|(1, \ell^X, 1), r_j, d_j|\star$ is built. The goal is to represent the color choice of each node v_i in graph G and check that there is no edge e_j in E where both nodes of the edge choose the same color.

Time segments: We segment time into $(m + 2)$ segments: a color choice segment $[0, 25n)$, m edge check segments along $[25n, 25(n + m))$ and a closing segment $[25(n+m), 25(n+2m)]$. Each node v_i has a time segment $[25i, 25(i+1))$ dedicated to its color choice - see Figure 2 - while each edge e_j is represented by time segment $[25(n + j), 25(n + j + 1))$ - see Figure 3.

Color loops: Color choices will be propagated to edge check segments using loops of properly coupled tasks. We define for each node v_i of G and each color $k \in \{0, 1, 2\}$ a color loop $\mathcal{L}_{i,k}$. The first time window of the loop is in the beginning of the color choice segment and the last time window is in the closing segment. This loop being ON corresponds to node v_i choosing color k. This is what will be used to connect color choices with our edge checks. The different parts of color loops $\mathcal{L}_{i,0}$, $\mathcal{L}_{i,1}$ and $\mathcal{L}_{i,2}$ can be seen in Figures 2, 3 and 4.

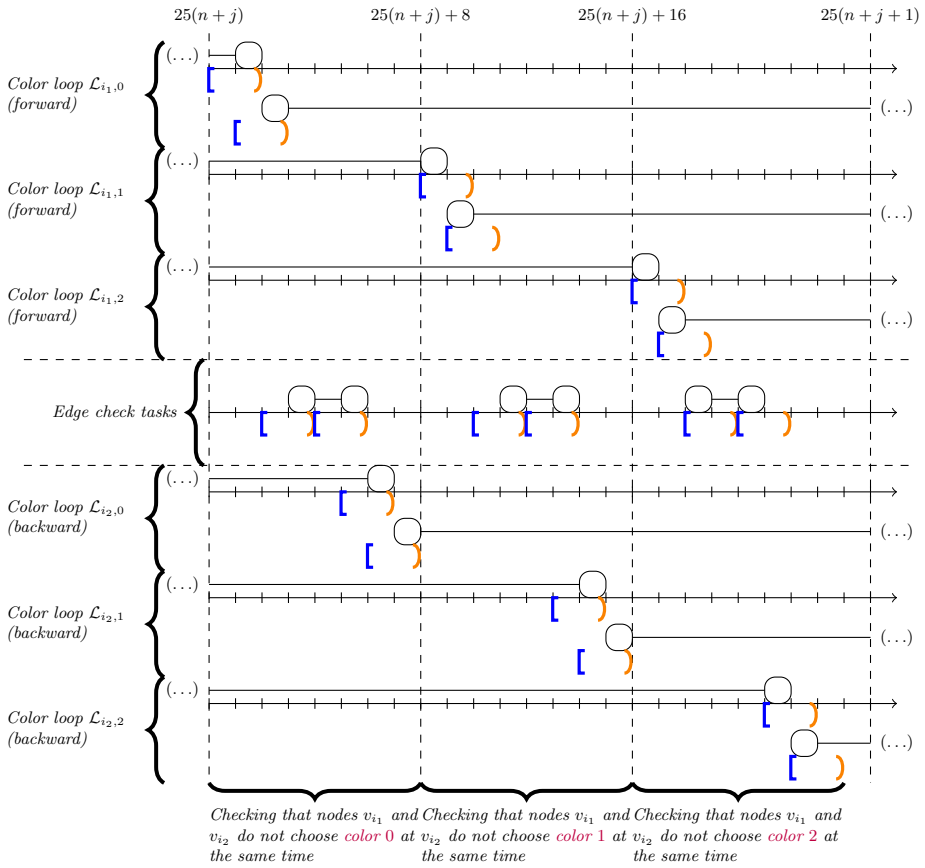

Fig. 3. The edge check segment corresponding to edge $e_j = \{v_{i_1}, v_{i_2}\}$ with $i_1 < i_2$ in the reduction with minimum delays. Here color loops $\mathcal{L}_{i_1,0}$ and $\mathcal{L}_{i_2,2}$ are ON and the other four are OFF.

Color choice tasks: For each node v_i a color choice is represented by a properly coupled task $(C_i(3), \ell^X, C'_i(3))$ of time window length 3. Then this coupled task is connected to the three corresponding color loops $\mathcal{L}_{i,k}$ using a few extra properly coupled tasks of time window length 2. The color choice is propagated the same way as in the color loops, i.e. with an alternation of proper delays and time windows overlaps. The goal is that if job $C_i(3)$ is scheduled at $r(C_i(3)) + k$ in a feasible schedule (with $k \in \{0, 1, 2\}$), then color loop $\mathcal{L}_{i,k}$ has to be in its ON state.

Figure 2 shows the color choice tasks and their interactions with the color loops in the minimum delays case. In this figure color task $C_i(3)$ starts at time $r(C_i(3)) + 1$, which indeed forces color loop $\mathcal{L}_{i,1}$ to be ON if we want a feasible schedule.

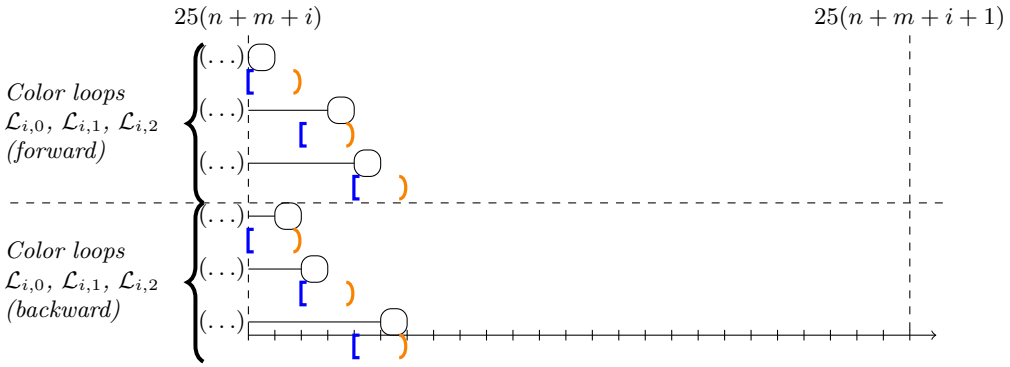

Fig. 4. The closing segment corresponding to node v_i in the reduction with minimum delays. Here color loop $\mathcal{L}_{i,1}$ is ON and the other two are OFF.

Edge check tasks: Recall that for each color $k \in \{0,1,2\}$ and each edge of G we defined a time segment where we check that both nodes of the edge do not have the same color k. For each edge $e_j = \{v_{i_1}, v_{i_2}\}$ with $i_1 < i_2$, an edge check for a given color $k \in \{0,1,2\}$ is represented by a properly coupled task $(E_j(k), \ell^X, E'_j(k))$ of time window length 2. This coupled task will overlap with the forward phase of loop $\mathcal{L}_{i_1,k}$ and the backward phase of loop $\mathcal{L}_{i_2,k}$. An example with minimum delays is given in Figure 3. Color loop $\mathcal{L}_{i_1,k}$ in its ON state forces coupled task $(E_j(k), \ell^X, E'_j(k))$ into one configuration, while color loop $\mathcal{L}_{i_2,k}$ in its ON state forces it into another configuration. This way having both loops ON makes it impossible to schedule these edge check tasks while at most one of them being ON is allowed.

For example in Figure 3 color loop $\mathcal{L}_{i_1,0}$ is ON and one of its forward tasks forces coupled task $(E_j(0), \ell^{min}, E'_j(0))$ to be scheduled at their release date plus one. This forces color loop $\mathcal{L}_{i_2,0}$ to be OFF in a feasible schedule - which it is in the figure. Similarly color loop $\mathcal{L}_{i_2,2}$ is ON and this time it is one of its backward tasks which forces coupled task $(E_j(2), \ell^{min}, E'_j(2))$ to be scheduled at their release date. This forces color loop $\mathcal{L}_{i_1,2}$ to be OFF in a feasible schedule - which it is in the figure.

3.2 NP-hardness of $1|(1, \ell^{min}, 1), r_j, d_j|\star$ with $\sigma = 2$

We build an instance of $1|(1, \ell^{min}, 1), r_j, d_j|\star$ with $\sigma = 2$. See Figures 2, 3 and 4 for an illustration of the color choice segment, an edge check segment and the closing segment respectively.

Forward phase $\mathcal{F}_{i,k}$: According to Definition 3 we define $\mathcal{F}_{i,k}$ a set of properly coupled tasks representing the forward phase of loop $\mathcal{L}_{i,k}$. Its purpose is to propagate the color choice of node v_i to the edge check segments of edges $e = \{v_i, v_j\}$ such that $i < j$.

Definition 4 (Forward phase $\mathcal{F}_{i,k}$). *Let $i \in [0, n-1]$ and $k \in \{0,1,2\}$. $\mathcal{F}_{i,k}$ contains the following properly coupled tasks:*

- *in the color choice segment: a properly coupled task $(F_{i,k}(0), \ell^{min}, F'_{i,k}(0))$:*
  - *case $\mathcal{F}_{i,0}$: $\langle 25i, 25i+9, 2\rangle$,*
  - *case $\mathcal{F}_{i,1}$: $\langle 25i+2, 25i+17, 2\rangle$,*
  - *case $\mathcal{F}_{i,2}$: $\langle 25i+4, 25i+22, 2\rangle$.*
- *If there is no edge $e = \{v_i, v_j\}$ such that $i < j$:*
  - *between the color choice segment and the the closing segment:*
    a properly coupled task $(F_{i,k}(1), \ell^{min}, F'_{i,k}(1))$
    $= \langle r(F'_{i,k}(0)) + 1, 25(n+m+i) + 2k, 2\rangle$
- *Else: let $j_1 < \ldots < j_r$ be the indices of the edges $e_{j_q} = \{v_i, v_j\}$ such that $i < j$.*
  - *between the color choice segment and the edge check segment of edge e_{j_1}:*
    a properly coupled task $(F_{i,k}(1), \ell^{min}, F'_{i,k}(1))$
    $= \langle r(F'_{i,k}(0)) + 1, 25(n+j_1) + 8k, 2\rangle$
  - *for each $q \in [1, r-1]$, between the edge check segments of edges e_{j_q} and $e_{j_{q+1}}$:*
    a properly coupled task $(F_{i,k}(q+1), \ell^{min}, F'_{i,k}(q+1))$
    $= \langle r(F'_{i,k}(q)) + 1, 25(n+j_{q+1}) + 8k, 2\rangle$.
  - *between the edge check segment of edge e_{j_r} and the closing segment:*
    a properly coupled task $(F_{i,k}(r+1), \ell^{min}, F'_{i,k}(r+1))$
    $= \langle r(F'_{i,k}(r)) + 1, 25(n+m+i) + 2k, 2\rangle$.

Backward task set $\mathcal{B}_{i,k}$: According to Definition 3 we define $\mathcal{B}_{i,k}$ a set of properly coupled tasks representing the backward phase of loop $\mathcal{L}_{i,k}$. Its purpose is to propagate the color choice of node v_i to the edge check segments of edges $e = \{v_i, v_j\}$ such that $i > j$.

Definition 5 (Backward task set $\mathcal{B}_{i,k}$). *Let $i \in [0, n-1]$ and $k \in \{0,1,2\}$. $\mathcal{B}_{i,k}$ contains the following properly coupled tasks:*

- *If there is no edge $e = \{v_i, v_j\}$ such that $i > j$:*
  - *between the color choice segment and the the closing segment:*
    a properly coupled task $(B_{i,k}(0), \ell^{min}, B'_{i,k}(0))$
    $= \langle 25i + 2k, 25(n+m+i) + 2k, 2\rangle$.
- *Else: let $j_1 < \ldots < j_r$ be the indices of the edges $e_{j_q} = \{v_i, v_j\}$ such that $i > j$.*
  - *between the color choice segment and the edge check segment of edge e_{j_1}:*
    a properly coupled task $(B_{i,k}(0), \ell^{min}, B'_{i,k}(0))$
    $= \langle 25i + 2k, 25(n+j_0) + 8k + 5, 2\rangle$,
  - *for each $q \in [1, r-1]$, between the edge check segments of edges e_{j_q} and $e_{j_{q+1}}$:*
    a properly coupled task $(B_{i,k}(q), \ell^{min}, B'_{i,k}(q))$
    $= \langle r(B'_{i,k}(q-1)) + 1, 25(n+j_q) + 8k + 5, 2\rangle$.
  - *between the edge check segment of edge e_{j_r} and the closing segment:*
    a properly coupled task $(B_{i,k}(r), \ell^{min}, B'_{i,k}(r))$
    $= \langle r(B'_{i,k}(r-1)) + 1, 25(n+m+i) + 2k, 2\rangle$.

Definition 6 (Color loop $\mathcal{L}_{i,k}$). *Let $i \in [0, n-1]$ and $k \in \{0,1,2\}$. Color loop $\mathcal{L}_{i,k}$ is the combination of properly coupled task sets $\mathcal{F}_{i,k}$ and $\mathcal{B}_{i,k}$.*

According to Lemma 2 we define the only two ways this loop can be scheduled:

1. *the ON state:*
   (a) *all tasks F in $\mathcal{F}_{i,k}$ are scheduled at time $r(F)+1$,*
   (b) *all tasks B in $\mathcal{B}_{i,k}$ are scheduled at time $r(B)$,*

2. *the OFF state:*
   (a) *all tasks F in $\mathcal{F}_{i,k}$ are scheduled at time $r(F)$,*
   (b) *all tasks B in $\mathcal{B}_{i,k}$ are scheduled at time $r(B)+1$.*

Definition 7 (Color choice task set $\mathcal{C}_i$). *Let $i \in [0, n-1]$. $\mathcal{C}_i$ is a set of four properly coupled tasks:*

- *a properly coupled task* $(C_i(0), \ell^{min}, C'_i(0)) = \langle 25i+6, 25i+8, 2\rangle$,
- *a properly coupled task* $(C_i(1), \ell^{min}, C'_i(1)) = \langle 25i+6, 25i+12, 2\rangle$,
- *a properly coupled task* $(C_i(2), \ell^{min}, C'_i(2)) = \langle 25i+14, 25i+16, 2\rangle$,
- *a properly coupled task* $(C_i(3), \ell^{min}, C'_i(3)) = \langle 25i+13, 25i+20, 3\rangle$.

Lemma 3. *Let $k \in \{0,1,2\}$. In any feasible schedule if color choice task $C_i(3)$ starts at time $r(C_i(3))+k$ then color loop $\mathcal{L}_{i,k}$ must be ON.*

Proof. We show that if color choice task $C_i(3)$ starts at time $r(C_i(3))+k$ then forward task $F'_{i,k}(0)$ must start at time $r(F'_{i,k}(1))+1$. Then Lemma 2 will imply that the whole loop $\mathcal{L}_{i,k}$ is ON.

- Case $k=2$: by the proper delay and the time window of length 3 job $C'_i(3)$ must be scheduled at time $r(C'_i(3))+2 = 25i+22$. However by Definition 4 this corresponds to $r(F'_{i,2}(0))$, with this task having a time window of length 2. Thus forward task $F'_{i,2}(0)$ must start at time $r(F'_{i,2}(0))+1$.
- Case $k=1$: color choice task $C_i(3)$ starts at time $r(C_i(3))+1 = 25i+14$ which corresponds to the release date of color choice task $C_i(2)$. Then with its time window of length 2 this task must be scheduled at time $r(C_i(2))+1$. By the proper delay and the time window of length 2 task $C'_i(2)$ must be scheduled at time $r(C'_i(2))+1 = 25i+17$. By Definition 4 corresponds to $r(F'_{i,1}(0))$. With this forward task having a time window of length 2 it must start at time $r(F'_{i,1}(0))+1$.
- Case $k=0$: color choice task $C_i(3)$ starts at time $r(C_i(3)) = 25i+13$ which corresponds to $r(C'_i(1))$. Then with its time window of length 2 this task must be scheduled at its release date. By the proper delay the other end $C_i(1)$ of this properly coupled task must also be scheduled at its release date. Then task $C_i(0)$, which has the same time window of length 2 as task $C_i(1)$, must be scheduled at its release date plus one. By the proper delay and the time window of length 2 the other end $C'_i(0)$ must be scheduled at time $r(C'_i(0))+1 = 25i+9$. By Definition 4 this corresponds to $r(F'_{i,0}(0))$. With this forward task having a time window of length 2 it must start at time $r(F'_{i,0}(0))+1$.

This concludes the proof of this lemma.

Definition 8 (Edge check task set $\mathcal{E}_j$). *Let $j \in [0, m-1]$. $\mathcal{E}_j$ is a set of three properly coupled tasks: for each $k \in \{0,1,2\}$ we have a properly coupled task $(E_j(k), \ell^{min}, E'_j(k)) = \langle 25(n+j)+8k+2, 25(n+j)+8k+4, 2\rangle$.*

Remark 1. This scheduling instance indeed has slack 2: there are n properly coupled tasks $(C_i(3), \ell^{min}, C'_i(3))$ with a time window length of 3 and all the other tasks have a time window length of 2.

Proposition 1. *G is 3-colorable if and only if there exists a feasible schedule for this instance of $1(1, \ell^{min}, 1), r_j, d_j|\star$.*

Proof. ($\implies$) Suppose we have $(c_0, \ldots, c_{n-1}) \in \{0,1,2\}^n$ a 3-coloring of G where vertex v_i has color c_i. We propose the following schedule:

- color loops $\mathcal{L}_{i,c_i}$ are ON while all the others are OFF
- color choice coupled task $(C_i(3), \ell^{min}, C'_i(3))$ are scheduled at their release date plus c_i. The other three are scheduled so that they do not interfere with the OFF color loops:
  - case $c_i = 0$: $(C_i(1), \ell^{min}, C'_i(1))$ and $(C_i(2), \ell^{min}, C'_i(2))$ at their release date, $(C_i(0), \ell^{min}, C'_i(0))$ at their release date plus one,
  - case $c_i = 1$: $(C_i(0), \ell^{min}, C'_i(0))$ at their release date, $(C_i(1), \ell^{min}, C'_i(1))$ and $(C_i(2), \ell^{min}, C'_i(2))$ at their release date plus one,
  - case $c_i = 2$: $(C_i(0), \ell^{min}, C'_i(0))$ and $(C_i(2), \ell^{min}, C'_i(2))$ at their release date, $(C_i(1), \ell^{min}, C'_i(1))$ at their release date plus one.
- given an edge $e_j = \{v_{i_1}, v_{i_2}\}$ with $i_1 < i_2$:
  - schedule edge check coupled task $(E_j(c_{i_1}), \ell^{min}, E'_j(c_{i_1}))$ at their release date plus one,
  - schedule edge check coupled task $(E_j(c_{i_2}), \ell^{min}, E'_j(c_{i_2}))$ at their release date,
  - schedule the remaining edge check coupled task of this edge check segment at their release date.

We show that this schedule is feasible. First we check the color choice segment. Let $i \in [0, n-1]$. For all three possible color choices c_i only one color choice task might interfere with a color loop: this task is scheduled at the release date of task $F'_{i,k}(0)$. However color loop $\mathcal{L}_{i,c_i}$ is ON, so $F'_{i,k}(0)$ is scheduled at its release date plus one and the machine constraint is fulfilled in this segment of the schedule. Now let $j \in [0, m-1]$. Let edge $e_j = \{v_{i_1}, v_{i_2}\}$ with $i_1 < i_2$. In the corresponding edge check segment only two color loops are ON: $\mathcal{L}_{i_1,c_{i_1}}$ the forward way and $\mathcal{L}_{i_2,c_{i_2}}$ the backward way. Thus in this edge check segment only two tasks from the color loops might interfere with an edge check task. Since $(c_0, \ldots, c_{n-1})$ is a 3-coloring we have $c_{i_1} \neq c_{i_2}$ and so two different edge check coupled tasks are involved. On the one hand let q_F be the index such that e_j is the q_F^{th} edge in the definition of the forward tasks. Then task $F_{i,c_{i_1}}(q_F+1)$ is scheduled at the release date of edge check task $E_j(c_{i_1})$ while the latter is scheduled at its release date plus one, so no interference there. On the other

hand let q_B be the index such that e_j is the q_B^{th} edge in the definition of the backward tasks. Then task $F'_{i,c_{i_2}}(q_B)$ is scheduled at the release date of edge check task $E'_j(c_{i_2})$ plus one while the latter is scheduled at its release date, so no interference there either. Thus the machine constraint is fulfilled in all edge check segments of the schedule. Finally all color loops are either ON or OFF so the machine constraint is also fulfilled in the closing segment. Thus the proposed schedule is feasible.

($\Longleftarrow$) Suppose we have a feasible schedule. For all $i \in [0, n-1]$ let $s_i \in \{0, 1, 2\}$ be such that $r(C_i(3)) + s_i$ is the starting time of color choice task $C_i(3)$. We show that $(s_0, \ldots, s_{n-1})$ is a 3-coloring of G. By contradiction suppose there is an edge $e_j = \{e_{i_1}, e_{i_2}\}$ such that $s_{i_1} = s_{i_2}$. Let $k = s_{i_1} = s_{i_2}$. Then by Lemma 3 both color loops $\mathcal{L}_{i_1,k}$ and $\mathcal{L}_{i_2,k}$ must be ON. Then all configurations of edge check task $(E_j(k), \ell^{min}, E'_j(k))$ are blocked: task $E_j(k)$ can only be scheduled at its release date plus one while task $E'_j(k)$ can only be scheduled at its release date. Then the proper minimum delay cannot be abided by. Thus the schedule is not feasible, which leads to a contradiction. Thus $(s_0, \ldots, s_{n-1})$ is indeed a 3-coloring of G.

This proves that $1|(1, \ell^{min}, 1), r_j, d_j|\star$ with $\sigma = 2$ is NP-hard, which concludes the para-NP-completeness proof of the corresponding parameterized problem.

Theorem 2. *$1|(1, \ell^{min}, 1), r_j, d_j|\star$ is para-NP-complete parameterized by slack σ.*

3.3 NP-hardness of $1|(1, \ell^{max}, 1), r_j, d_j|\star$ with $\sigma = 2$

We build an instance of $1|(1, \ell^{max}, 1), r_j, d_j|\star$ with $\sigma = 2$. See Figures 5, 6 and 7 for an illustration of the color choice segment, an edge check segment and the closing segment respectively. The reduction is very similar to the minimum delays case, except most jobs have their window slightly shifted in order to propagate color choices with maximum delays instead of minimum ones.

Definition 9 (Forward task set $\mathcal{F}_{i,k}$). *Let $i \in [0, n-1]$ and $k \in \{0, 1, 2\}$. $\mathcal{F}_{i,k}$ contains the following properly coupled tasks:*

- *in the color choice segment: a properly coupled task $(F_{i,k}(0), \ell^{max}, F'_{i,k}(0))$:*
  - *case $\mathcal{F}_{i,0}$: $\langle 25i, 25i + 13, 2\rangle$,*
  - *case $\mathcal{F}_{i,1}$: $\langle 25i + 2, 25i + 18, 2\rangle$,*
  - *case $\mathcal{F}_{i,2}$: $\langle 25i + 4, 25i + 22, 2\rangle$.*
- *If there is no edge $e = \{v_i, v_j\}$ such that $i < j$:*
  - *between the color choice segment and the the closing segment:*
    a properly coupled task $(F_{i,k}(1), \ell^{max}, F'_{i,k}(1))$
    $= \langle r(F'_{i,k}(0)) - 1, 25(n + m + i), 2\rangle$
- *Else: let $j_1 < \ldots < j_r$ be the indices of the edges $e_{j_q} = \{v_i, v_j\}$ such that $i < j$.*
  - *between the color choice segment and the edge check segment of edge e_{j_1}:*
    a properly coupled task $(F_{i,k}(1), \ell^{max}, F'_{i,k}(1))$
    $= \langle r(F'_{i,k}(0)) - 1, 25(n + j_1) + 8k + 2, 2\rangle$

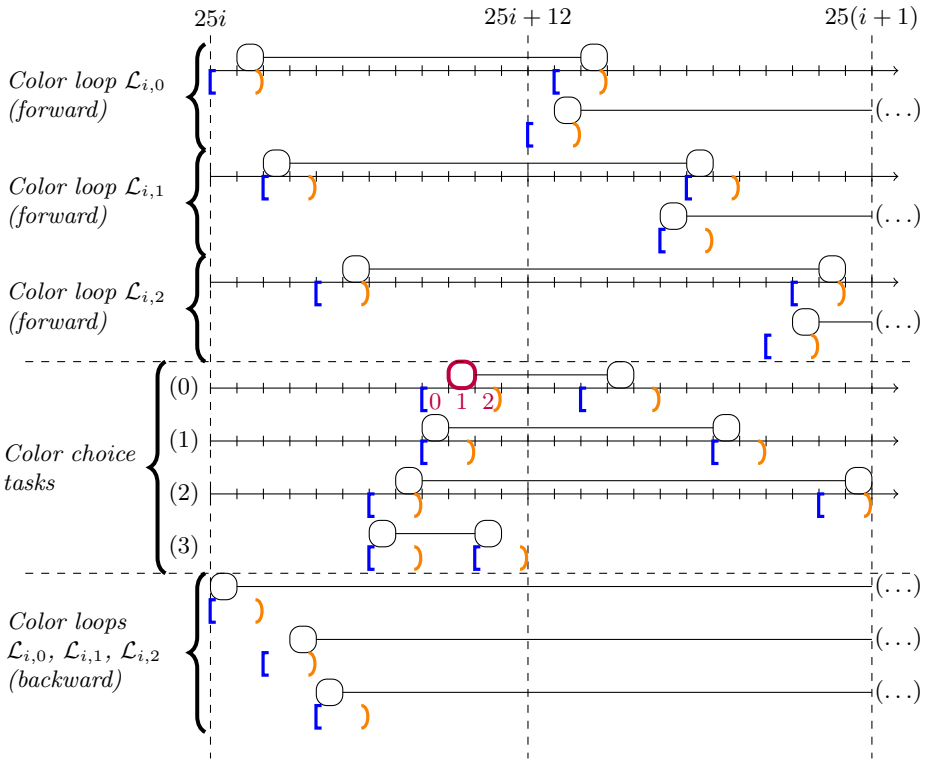

Fig. 5. The color choice segment corresponding to node v_i in the reduction with maximum delays. Color loop $\mathcal{L}_{i,1}$ is ON and the other two are OFF.

- *for each* $q \in [1, r-1]$*, between the edge check segments of edges* e_{j_q} *and* $e_{j_{q+1}}$*:*
  a properly coupled task $(F_{i,k}(q+1), \ell^{max}, F'_{i,k}(q+1))$
  $= \langle r(F'_{i,k}(q)) - 1, 25(n + j_{q+1}) + 8k + 2, 2\rangle$
- *between the edge check segment of edge* e_{j_r} *and the closing segment:*
  a properly coupled task $(F_{i,k}(r+1), \ell^{max}, F'_{i,k}(r+1))$
  $= \langle r(F'_{i,k}(r)) - 1, 25(n + m + i) + 2k, 2\rangle$.

Definition 10 (Backward task set $\mathcal{B}_{i,k}$). *Let* $i \in [0, n-1]$ *and* $k \in \{0, 1, 2\}$. $\mathcal{B}_{i,k}$ *contains the following properly coupled tasks:*

- *If there is no edge* $e = \{v_i, v_j\}$ *such that* $i > j$*:*
  - *between the color choice segment and the the closing segment:*
    a properly coupled task $(B_{i,k}(0), \ell^{max}, B'_{i,k}(0))$
    $= \langle 25i + 2k, 25(n + m + i) + 2k, 2\rangle$.
- *Else: let* $j_1 < \ldots < j_r$ *be the indices of the edges* $e_{j_q} = \{v_i, v_j\}$ *such that* $i > j$.

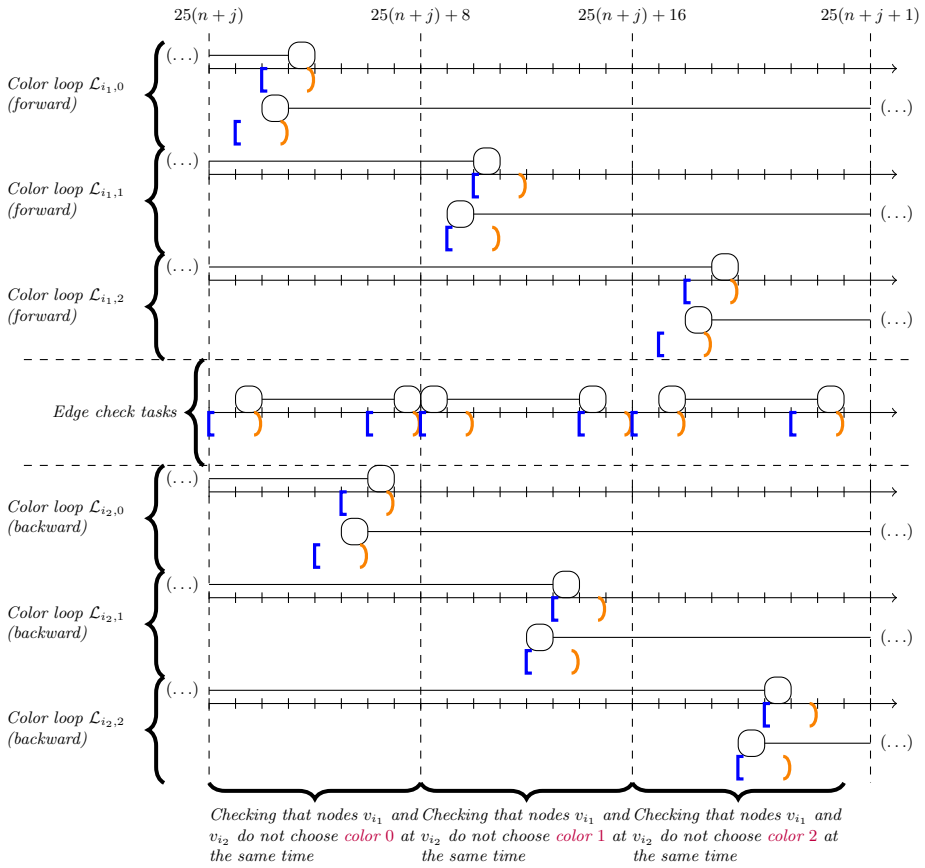

Fig. 6. The edge check segment corresponding to edge $e_j = \{v_{i_1}, v_{i_2}\}$ with $i_1 < i_2$ in the reduction with maximum delays. Here color loops $\mathcal{L}_{i_1,1}$ and $\mathcal{L}_{i_2,0}$ are ON and the other four are OFF.

- *between the color choice segment and the edge check segment of edge* e_{j_1}*:*
  a properly coupled task $(B_{i,k}(0), \ell^{max}, B'_{i,k}(0))$
  $= \langle 25i + 2k, 25(n + j_1) + 8k + 5, 2\rangle$,
- *for each* $q \in [1, r-1]$*, between the edge check segments of edges* e_{j_q} *and* $e_{j_{q+1}}$*:*
  a properly coupled task $(B_{i,k}(q), \ell^{max}, B'_{i,k}(q))$
  $= \langle r(B'_{i,k}(q-1)) - 1, 25(n + j_{s+1}) + 8k + 5, 2\rangle$.
- *between the edge check segment of edge* e_{j_r} *and the closing segment:*
  a properly coupled task $(B_{i,k}(r), \ell^{max}, B'_{i,k}(r))$
  $= \langle r(B'_{i,k}(r-1)) - 1, 25(n + m + i) + 2k, 2\rangle$.

Definition 11 (Color loop $\mathcal{L}_{i,k}$). *Let* $i \in [0, n-1]$ *and* $k \in \{0, 1, 2\}$*. Color loop* $\mathcal{L}_{i,k}$ *is the combination of properly coupled task sets* $\mathcal{F}_{i,k}$ *and* $\mathcal{B}_{i,k}$*.*

We define the following states:

1. *the ON state:*
   (a) *all tasks F in $\mathcal{F}_{i,k}$ are scheduled at time $r(F)$,*
   (b) *all tasks B in $\mathcal{B}_{i,k}$ are scheduled at time $r(B)+1$,*

2. *the OFF state:*
   (a) *all tasks F in $\mathcal{F}_{i,k}$ are scheduled at time $r(F)+1$,*
   (b) *all tasks B in $\mathcal{B}_{i,k}$ are scheduled at time $r(B)$.*

Definition 12 (Color choice task set $\mathcal{C}_i$). *Let $i \in [0, n-1]$. $\mathcal{C}_i$ is a set of four properly coupled tasks:*

- *a properly coupled task $(C_i(0), \ell^{max}, C'_i(0)) = \langle 25i+8, 25i+14, 3\rangle$,*
- *a properly coupled task $(C_i(1), \ell^{max}, C'_i(1)) = \langle 25i+8, 25i+19, 2\rangle$,*
- *a properly coupled task $(C_i(2), \ell^{max}, C'_i(2)) = \langle 25i+6, 25i+23, 2\rangle$,*
- *a properly coupled task $(C_i(3), \ell^{max}, C'_i(3)) = \langle 25i+6, 25i+10, 2\rangle$.*

Lemma 4. *Let $k \in \{0,1,2\}$. In any feasible schedule if color choice task $C_i(0)$ starts at time $r(C_i(0))+k$ then color loop $\mathcal{L}_{i,k}$ must be ON.*

Proof. This is proved in the same fashion as Lemma 3.

Definition 13 (Edge check task set $\mathcal{E}_j$). *Let $j \in [0, m-1]$. $\mathcal{E}_j$ is a set of three properly coupled tasks: for each $k \in \{0,1,2\}$ we have a properly coupled task $(E_j(k), \ell^{max}, E'_j(k)) = \langle 25(n+j)+8k, 25(n+j)+8k+6, 2\rangle$.*

Remark 2. This scheduling instance indeed has slack 2: there are n properly coupled tasks $(C_i(0), \ell^{max}, C'_i(0))$ with a time window length of 3 and all the other tasks have a time window length of 2.

Proposition 2. *G is 3-colorable if and only if there exists a feasible schedule for this instance of $1|(1, \ell^{max}, 1), r_j, d_j|\star$.*

Proof. This is proved in the same fashion as Lemma 1.

This proves that $1|(1, \ell^{max}, 1), r_j, d_j|\star$ with $\sigma = 2$ is NP-hard, which concludes the para-NP-completeness proof of the corresponding parameterized problem.

Theorem 3. *$1|(1, \ell^{max}, 1), r_j, d_j|\star$ is para-NP-complete parameterized by slack σ.*

3.4 NP-hardness of $1|(1, \ell^{ex}, 1), r_j, d_j|\star$ with $\sigma = 2$

We build an instance of $1|(1, \ell^{ex}, 1), r_j, d_j|\star$ with $\sigma = 2$. One can simply take the reduction with minimum delays and replace the minimum delays with exact ones. Then the whole reduction works the same way: color loops still have the same two possible schedules, each color choice is propagated through color choice tasks and color loops the exact same way, and the ON color loops force the edge check tasks to be scheduled at the exact same configurations.

This proves that $1|(1, \ell^{ex}, 1), r_j, d_j|\star$ with $\sigma = 2$ is NP-hard, which concludes the para-NP-completeness proof of the corresponding parameterized problem.

Corollary 1. *$1|(1, \ell^{ex}, 1), r_j, d_j|\star$ is para-NP-complete parameterized by slack σ.*

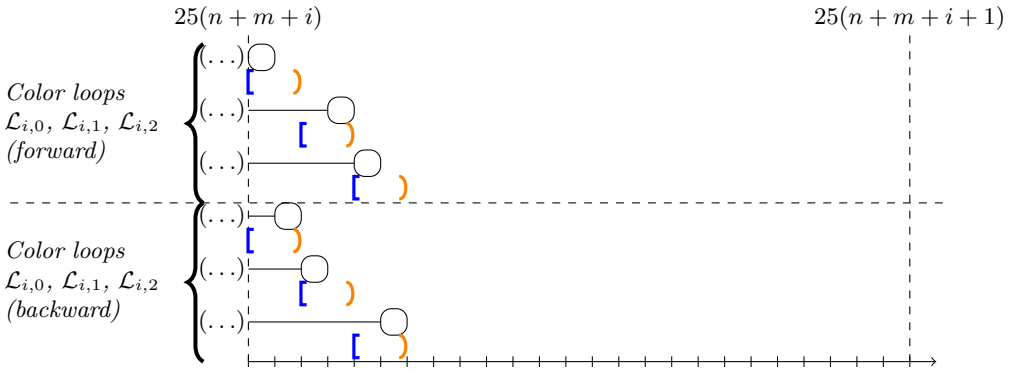

Fig. 7. The closing segment corresponding to node v_i in the reduction with maximum delays. Here color loop $\mathcal{L}_{i,1}$ is OFF and the other two are ON.

3.5 Implications to parameter μ

With Theorem 1 we know that our three results of para-NP-completeness with slack σ imply para-NP-completeness with pathwidth as the parameter instead. In fact upon closer inspection of our reductions, there were never more than two time windows overlapping at any time. This leads to the following result:

Corollary 2. *The problems* $1|(1,\ell^X,1),r_j,d_j|\star$ *with* $X \in \{min, max, ex\}$ *and* $\mu = 1$ *are (strongly)* NP*-hard. As a consequence they are all para-*NP*-complete parameterized by* μ*.*

4 Polynomial-time solvability with $\sigma = 1$

In this section delays can be exact, minimum or maximum. We prove polynomial-time solvability of the scheduling problem $1|prec, p_i = 1, \ell_{ij}, r_i, d_i, \sigma = 1|\star$ by reducing it to 2-SAT. We achieve this in two steps: first we reduce from a slightly more restricted scheduling problem with proper delays and all time windows of length 2, with a straightforward reduction to 2-SAT. Then we show how to convert a general instance of this problem into a restricted one equivalent to it in terms of feasibility.

4.1 Reduction with time windows of length exactly 2 and proper delays

Let us consider an instance of our restricted problem. The set of n jobs is denoted by $\mathcal{T}$. Each job J must be performed in a time window of length 2 $[r(J), r(J)+2)$. We consider an acyclic graph G of precedence constraints with proper delays, so that for each arc $a = (J_i, J_j)$ in G, its delay $\ell_a = r(J_j) - r(J_i) - 1$.

We build a boolean formula in conjunctive normal form with only clauses of two literals. This formula will encode our scheduling instance and we want it to be satisfiable if and only if the scheduling instance is feasible.

We have n boolean variables x_i, one per job in the scheduling instance. All jobs have exactly two possible starting times, which correspond to the two possible values of each boolean variable. In particular we want to have:

$$x_i = \begin{cases} 0 & \text{if job } J_i \text{ is scheduled at time } r(J_i) \\ 1 & \text{if job } J_i \text{ is scheduled at time } r(J_i) + 1 \end{cases} \tag{4}$$

We now describe the clauses in our boolean formula. For the sake of better understanding clauses will be presented as implications of the form $(x \implies y)$ which are indeed equivalent to clauses of two literals $(\neg x \vee y)$ in classical logic. We distinguish two types of clauses: clauses from machine constraints and clauses from precedence constraints. The former will depend on the length of the overlap of time windows while the latter will take into account the nature of the delay (exact, minimum or maximum).

Machine constraints: There are two possibilities, which are described below. Consider two jobs $J_i, J_j \in \mathcal{T}$ and without loss of generality assume that $r(J_i) \geq r(J_j)$. If their two time windows are disjoint, no constraint is needed. Otherwise, there may be two cases:

1. The two jobs have the same time windows, so $r(J_i) = r(J_j)$. Two symmetric clauses are then added $(x_i \implies \neg x_j)$ and $(x_j \implies \neg x_i)$ in order to ensure that if J_i is scheduled at $r(J_i)$ then J_j is scheduled at $r(J_i) + 1$, and conversely.
2. The jobs time windows overlap on one time unit, so $r(J_i) = r(J_j) + 1$. We must ensure that if J_j is scheduled at $r(J_j) + 1$ J_i is not scheduled at the same time: $(x_j \implies x_i)$.

Precedence constraints : Consider a precedence constraint $a = (J_i, J_j)$ with proper delay $\ell_a = r(J_j) - r(J_i) - 1$. The clauses depend on the nature of the delay (minimum, maximum or exact).

- In case of a minimum delay: $x_i \implies x_j$ (if J_i is right shifted in its time window, so is J_j)
- In case of a maximum delay: $\neg x_i \implies \neg x_j$ (if J_i is left shifted in its time window, so is J_j)
- In case of an exact delay, the two clauses associated to minimum and maximum delays are added.

If I is an instance of the scheduling problem, we denote by $SAT(I)$ the $2-SAT$ associated instance as described above.

We can now state our lemma:

Lemma 5. *An instance I of the one machine scheduling problem with precedence constraints, proper delays and time windows of length 2 is feasible if and only if $SAT(I)$ is satisfiable.*

Proof. ($\implies$) Let us consider a feasible schedule for the instance I and the instantiation of variables described in Equation (4). As the machine constraint is satisfied, no two jobs are scheduled at the same time. So if $r(J_i) = r(J_j)$ and J_i is scheduled at $r(J_i)$, then J_j is scheduled at $r(J_i)+1$ (and symmetrically). So we cannot have $x_i \wedge x_j$ or $\neg x_i \wedge \neg x_j$. Similarly if $r(J_j) = r(J_i)+1$, then J_i and J_j are not scheduled both at $r(J_i)+1$ so $x_i \wedge \neg x_j$ is false (and thus the implication is true). Now for a precedence constraint a with a proper delay, depending on the kind of delay, all the implications are true with similar arguments. For example for a minimum delay, if J_i is scheduled at $r(J_i) + 1$ then J_j is scheduled at $r(J_i) + 1 + 1 + \ell_a = r(J_j) + 1$, so $x_i \implies x_j$ is true.

($\impliedby$) Let us consider a feasible instantiation of variables fo $SAT(I)$. We schedule the jobs according to Equation (4). Let us verify that the precedence constraints are satisfied. Let $a = (J_i, J_j)$ be a precedence constraint. Assume first that this is a minimum delay. Then if $x_i = 0$, J_j can be scheduled in both positions. Otherwise, if $x_i = 1$ then we know by the corresponding clause that $x_j = 1$, so that J_j is scheduled at $r(J_j)+1 = r(J_i)+1+1+\ell_a$, and the precedence constraint is satisfied. Similar arguments hold for maximum and exact delays. Now assume that there would be two jobs J_i, J_j scheduled at the same time. These two jobs would have overlapping intervals, and assume that $r(J_i) \geq r(J_j)$. Assume that both jobs are scheduled at $r(J_i)$. Then if $r(J_j) = r(J_j)$ we would have $x_i = x_j = 0$ which is impossible since $x_j \vee \neg x_i$, so that $\neg x_j \vee x_i = 1$. Similarly if $r(J_i) = r(J_j) + 1$ then $x_i = 0, x_j = 1$ which is impossible since the clause $(x_j \implies x_i)$ is satisfied. Similar arguments hold assuming the two jobs are performed at $r(J_i) + 1$.

4.2 Preprocessing from time windows of length at most 2 and general delays

Starting from a scheduling instance of $1|prec(\ell_{i,j}), p_j = 1, r_j, d_j \leq r_j + 2|\star$ there are two missions in this preprocessing step:

1. dealing with non-proper delays.
2. removing jobs with a time window of length 1,

1. Dealing with non-proper delays: Consider a precedence relation $a = (J_i, J_j)$ with delay ℓ_a. Let us consider the case of minimum delays first.

If $r(J_j) + 1 < r(J_i) + 1 + \ell_a$ then no feasible schedule exist: even if J_i is scheduled as early as possible, the completion time of J_i plus the minimum delay is greater than all possible execution times for J_j. Let us assume that $r(J_j) - r(J_i) \geq \ell_a$. If $r(J_j) - r(J_i) = \ell_a$ then there is only one possibility to fulfill the precedence constraint: J_i should be scheduled at $r(J_i)$ and job J_j at $r(J_j) + 1$. So the time windows of both jobs can be reduced to time windows of length 1: $d(J_i) = r(J_i) + 1, r(J_j) = d(J_j) - 1$. The precedence constraint is then useless. Now assume that $r(J_j) - r(J_i) - 1 \geq \ell_a$. If the inequality is strict, then in any schedule fulfilling the time windows, this constraint will be satisfied: indeed if J_i is scheduled at $r(J_i) + 1$, then J_j can be scheduled at r_j and the

precedence constraint still holds. Hence in this case, the precedence constraint can be removed without loss of generality. This proves that in case of minimum non proper delays, either we can conclude of infeasibility, or reduce the time windows, or remove the precedence constraint.

Let us now consider the case of maximum delays. Symmetrically, If $r(J_j) > r(J_i) + 2 + \ell_a$ then no feasible schedule exist. If $r(J_j) = r(J_i) + 2 + \ell_a$, then J_i must be scheduled at $r(J_i) + 1$ while J_j must be scheduled at $r(J_j)$, so the time windows of the two jobs can be reduced and the precedence constraint removed. If $r(J_j) < r(J_i) + 1 + \ell_a$ the precedence constraint can be removed: it is fulfilled even if J_i is scheduled at $r(J_i)$ and J_j at $r(J_j) + 1$. Hence only proper delays are of interest here.

Finally, in the case of exact delays, which combine the constraints of both minmum and maximum delays, the same property holds. Given an instance I of $1|prec(\ell_{i,j}), p_j = 1, r_j, d_j \leq r_j + 2|\star$, either we can conclude of infeasibility, or there is an equivalent instance $prop(I)$ of the problem with proper delays.

2. Removing jobs with a time window of length 1: such a job J_j has no other option and books an entire time position for itself since there is a single machine. Thus other jobs will never been scheduled at that time and this time position can be removed in the time windows of the other jobs. If some time window becomes empty, the instance is unfeasible. The iterative application of such transformation of the instance I is called $Prune(I)$. Each time such a transformation is made an equivalent instance with proper delays $Prop(Prune(I))$ can be computed.

Consider now a precedence constraint $a = (J_i, J_j)$ with delay $\ell_a = r(J_j) - r(J_i) - 1$ as we assumed proper delays.

If the time window of both jobs J_i, J_j is of length 1, the precedence constraint is always satisfied, so it can be removed from the instance.

If the time window of J_i is of length 2 and the other of J_j is of length 1, then in the case of a minimum delay or exact delay, the only valid position for J_i is left shifted, so that its interval can be reduced to $[r(J_i), r(J_i) + 1)$ and the precedence constraint removed. In the case of a maximum delay, both positions for J_i will satisfy the constraint, so the precedence constraint can be removed.

Symmetrically, if the time window of J_i is of length 1 and the other of J_j is of length 2, then similar arguments lead to either reduce the time window of J_j in the case of maximum or exact delays, and in any case to remove the precedence constraint.

Such a transformation of instance I based on the precedence constraints is called $Prec(I)$

We can then iteratively apply the operations $Prune$, $Prop$ and $Prec$ either to detect unfeasibility, or to generate an instance such that:

- All delays are proper delays
- No time window of length 1 is included in another time window
- No job with time window of length 1 has a predecessor nor a successor in the precedence graph.

Such transformation can be done in polynomial time (at each step at least one time window is reduced or one arc is removed) Then, the jobs with time window of length 1 can be removed from the instance, because the feasibility does not rely on them anymore.

This leads to our result:

Lemma 6. *Considering an instance I of $1|prec(\ell_{i,j}), p_j = 1, r_j, d_j \leq r_j + 2|\star$ there is a polynomial algorithm that either indicates that instance I is unfeasible, or computes an instance I' in which all delays are proper delays and all time windows are of length 2.*

From Lemma 6 we deduce the main result of this section:

Theorem 4. *$1|prec(\ell_{i,j}), p_j = 1, r_j, d_j \leq r_j + 2|\star$ is polynomially solvable*

Proof. By applying the transformation described in Lemma 6 we can either detect unfeasibility or generate an equivalent instance with proper delays and time windows of length 2. By Lemma 5, the problem is polynomially solvable.

5 Conclusion

In this paper we established a relation between the pathwidth and the slack, which are two parameters commonly considered to analyze the parameterized complexity of scheduling problems with time windows. This relation leads to new complexity results: FPT with the slack and para-NP-completeness proofs with the pathwidth. We also proved that scheduling coupled tasks on a single machine with time windows is para-NP-hard parameterized by the slack for either minimum, maximum or exact precedence delays. Lastly we established that the problem with time windows of length at most 2 and precedence delays - minimum, maximum or exact - can be solved in polynomial time.

Looking ahead, we believe that the relations between the parameters usually considered for scheduling problems should be further investigated. In fact the objective would be to replicate the method used in this paper - where a negative complexity result for the slack induced a negative one for the pathwidth - for other existing scheduling parameters. We also feel that the slack should be investigated beyond feasibility. It is likely that some of the existing FPT results could be extended to several optimization criteria. Finally finding a frontier subproblem with a FPT algorithm for the slack while hard when parameterized by the pathwidth would complete the analysis of the relation between the two parameters.

References

1. Baart, R., de Weerdt, M., He, L.: Single-machine scheduling with release times, deadlines, setup times, and rejection. European Journal of Operational Research **291**(2), 629–639 (2021), number: 2 Publisher: Elsevier

2. Bessy, S., Giroudeau, R.: Parameterized complexity of a coupled-task scheduling problem. Journal of Scheduling **22**(3), 305–313 (Jun 2019). https://doi.org/10.1007/s10951-018-0581-1, https://doi.org/10.1007/s10951-018-0581-1
3. van Bevern, R., Bredereck, R., Bulteau, L., Komusiewicz, C., Talmon, N., Woeginger, G.J.: Precedence-constrained scheduling problems parameterized by partial order width. In: Kochetov, Y., Khachay, M., Beresnev, V., Nurminski, E., Pardalos, P. (eds.) Discrete Optimization and Operations Research. pp. 105–120. Springer International Publishing, Cham (2016)
4. Bodlaender, H.L., van der Wegen, M.: Parameterized complexity of scheduling chains of jobs with delays. In: 15th International Symposium on Parameterized and Exact Computation (IPEC) (2020)
5. Chen, B., Zhang, X.: Scheduling coupled tasks with exact delays for minimum total job completion time. J. Sched. **24**(2), 209–221 (2021). https://doi.org/10.1007/s10951-020-00668-1, https://doi.org/10.1007/s10951-020-00668-1
6. Downey, R., Fellows, M.: Parameterized complexity. Springer (1999)
7. Downey, R.G., Fellows, M.R., Regan, K.W.: Descriptive complexity and the W hierarchy. In: Proof Complexity and Feasible Arithmetics. pp. 119–134 (1996)
8. Flum, J., Grohe, M.: Parameterized complexity theory. Springer (1998)
9. Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.R.: Optimization and approximation in deterministic sequencing and scheduling: a survey. In: Annals of discrete mathematics, vol. 5, pp. 287–326. Elsevier (1979)
10. Hanen, C., Mallem, M., Munier-Kordon, A.: Parameterized complexity of single-machine scheduling with precedence, release dates and deadlines. In: 15th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP) (2022)
11. Hanen, C., Munier-Kordon, A.: Fixed-Parameter tractability of scheduling dependent typed tasks subject to release times and deadlines. Submitted (2021)
12. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of computer computations, pp. 85–103. Springer (1972)
13. Khatami, M., Salehipour, A., Cheng, T.: Coupled task scheduling with exact delays: Literature review and models. European Journal of Operational Research **282**(1), 19–39 (2020). https://doi.org/https://doi.org/10.1016/j.ejor.2019.08.045, https://www.sciencedirect.com/science/article/pii/S0377221719307155
14. Lawler, E.: Optimal sequencing of a single machine subject to precedence constraints. Management Sci. **19**, 544–546 (1973)
15. Lenstra, J., Rinnooy Kan, A., Brucker, P.: Complexity of machine scheduling problems. Ann. of Discrete Math. **1**, 343–362 (1977)
16. Mallem, M., Hanen, C., Munier-Kordon, A.: Parameterized complexity of a parallel machine scheduling problem. In: 17th International Symposium on Parameterized and Exact Computation (IPEC) (2022)
17. Martel, C.U.: Preemptive scheduling with release times, deadlines, and due times. J. ACM **29**(3), 812–829 (1982). https://doi.org/10.1145/322326.322337, https://doi.org/10.1145/322326.322337
18. Mnich, M., Wiese, A.: Scheduling and fixed-parameter tractability. Mathematical Programming **154**(1-2), 533–562 (Dec 2015). https://doi.org/10.1007/s10107-014-0830-9, http://link.springer.com/10.1007/s10107-014-0830-9
19. Munier Kordon, A.: A fixed-parameter algorithm for scheduling unit dependent tasks on parallel machines with time windows. Discrete Applied Mathematics **290**, 1–6 (2021)

20. Yu, W., Hoogeveen, H., Lenstra, J.K.: Minimizing Makespan in a Two-Machine Flow Shop with Delays and Unit-Time Operations is NP-Hard. Journal of Scheduling **7**(5), 333–348 (Sep 2004). https://doi.org/10.1023/B:JOSH.0000036858.59787.c2, https://doi.org/10.1023/B:JOSH.0000036858.59787.c2